Lab 9 - NIDS/NIPS and Web Proxy Analysis

University of Maryland Baltimore County

Presented to: (professor's name removed)

April 17, 2020

Table of Content

Introduction

While working with Division of IT at University of Maryland Baltimore County, for their Digital Forensics and Incident Response team, I have been tasked to a) Perform log analysis on IDS/IPS alerts (Snort) to become familiar with alert formats and using them to investigate potentially malicious traffic and b) Perform log analysis on Proxy logs (Squid) to become familiar with proxy log formats, and how to review them to investigate web related traffic. The methodology adopted for the analysis of the given alert, tcpdump.log, snort.conf, access.log and store.log files is done by using a combination of tools and linux terminal commands– wireshark, bless hex editor, grep command and MS Excel. This task encompasses in detail analysis of the logs and alerts generated by Network Intrusion Protection and Detection Systems (NIDS/NIPS).

In this fictitious scenario, you'll be provided IDS/IPS alert data and proxy logs in an effort to assist an investigation.

In his quest to save the planet, Inter0ptic has started a credit card number recycling program. “Do you have a database filled with credit card numbers, just sitting there collecting dust? Put that data to good use!” he writes on his website. “Recycle your company’s used credit card numbers! Send us your database, and we’ll send YOU a check.” For good measure, .Inter0ptic decides to add some bells and whistles to the site too.

Meanwhile…. MacDaddy Payment Processor has deployed Snort NIDS sensors to detect an array of anomalous events, bot inbound and outbound. An alert was logged at 08:01:45 on 5/18/11 concerning an inbound chunk of executable code sent to port 80/tcp for inside host 192.168.1.169 from external host 172.16.16.218.

Alert:

[] [1:10000648:2] SHELLCODE x86 NOOP [] [Classification: Executable code was detected][Priority: 1] 5/18-08:01:45.591840 172.16.16.218:80 -> 192.168.1.169:2493 TCP TTL: 63 TOS:0x0 ID:53309 IpLen: 20 DgmLen: 1127 DF AP Seq: 0x1B2C3517 Ack: 0x9F9E0666 Win: 0x1920 TcpLen: 20

Network:

Internal network: 192.168.1.0/24 DMZ network: 10.1.1.0/24 “Internet”: 172.16.16.0/24 Other domains and subnets of interest include: .evl – a top level domain used by Evil Systems Example.com – MacDaddy Payment Processor local domain

The security staff at MacDaddy Payment Processor collects the Snort alerts for the day in question and preserves the “tcpdump.log” file that corresponds with the alerts. At your request, they also gather the relevant Snort sensor’s config and rules. You are provided with the following files containing data to analyze:

- **Alert** – A text file containing the Snort sensor’s default “alert” output, including the alert of interest above
- **Tcpdump.log** – A libpcap generated file that contains full-content packet captures of the packets involved in the events summarized in the above alert file.
- **Snort.conf** – A text file containing the configuration description of the Snort sensor, including the rules
- **Rules** – A folder containing the Snort rules that were in use by the sensor, as included by the configuration above.

All files are located under the /cases/Evidence/ within the IDS/IPS Alert Analysis folder.

Analysis

1. IDS/IPS Alert Background (Part 1)

1.1 Examine the alert's data to understand the logistical context

Figure 1.1.1, we analyse the alert and note: Source IP: 172.16.16.218 (Source Port :80) and Destination IP: 192.168.1.169:2493, Packet Delivery ID : 53309, Snort ID: 10000648, IPlink : 20 and Datagram link : 1127

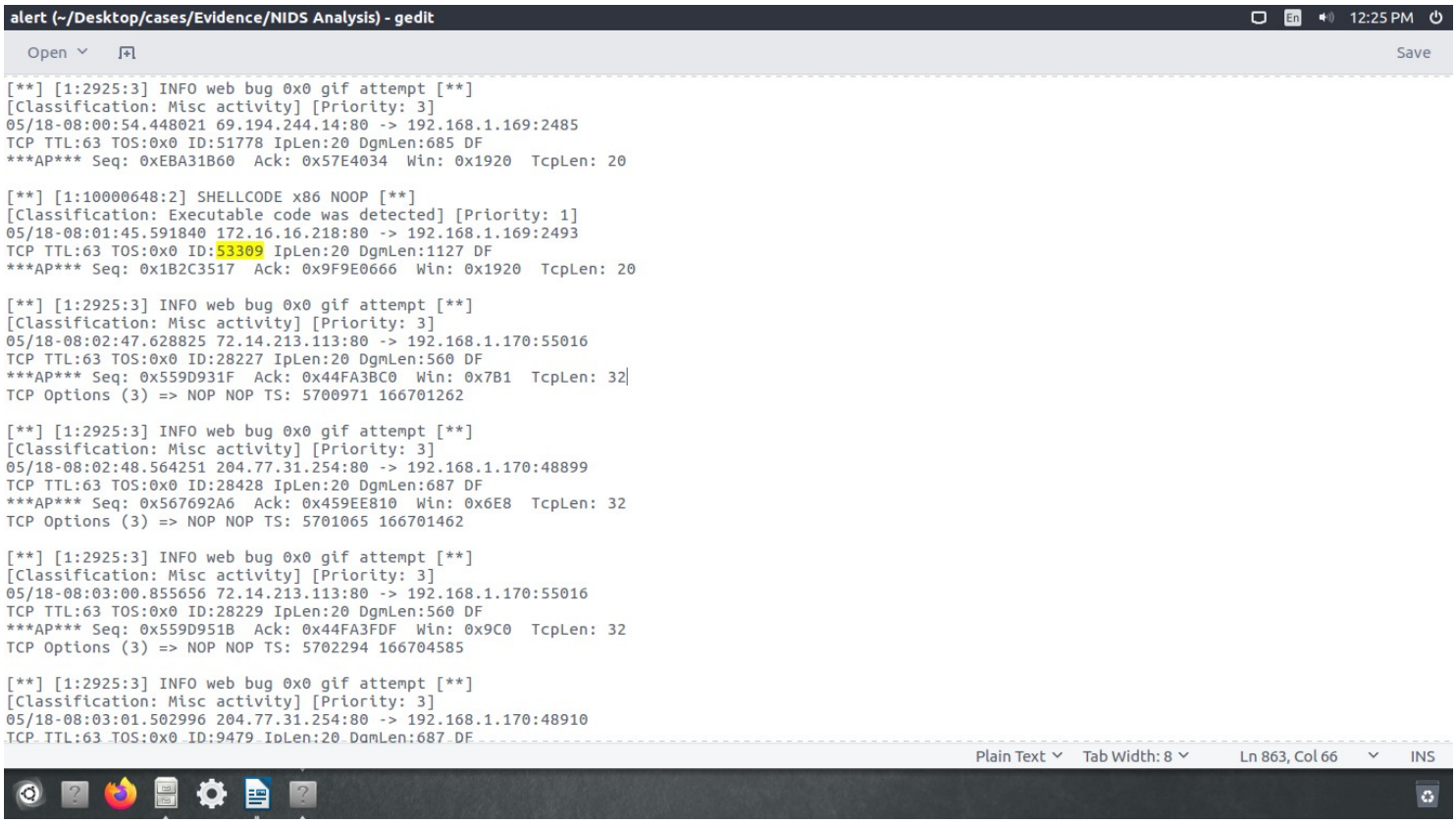

Figure 1.1.1

1.2 Compare the alert to the rule, in order to better understand WHAT it has been built to detect:

Figure 1.2.1, suing the snort ID we use **cd /home/sansforensics/Desktop/cases/Evidence/"NIDS Analysis"/rules** and then **grep -r sid:10000648 *.rules** to find the **shellcode-**detect among a list of .rules files and understand more about alert and the rule using the sid and find that **local.rule** file mentions the **classtype : shellcode-detect**

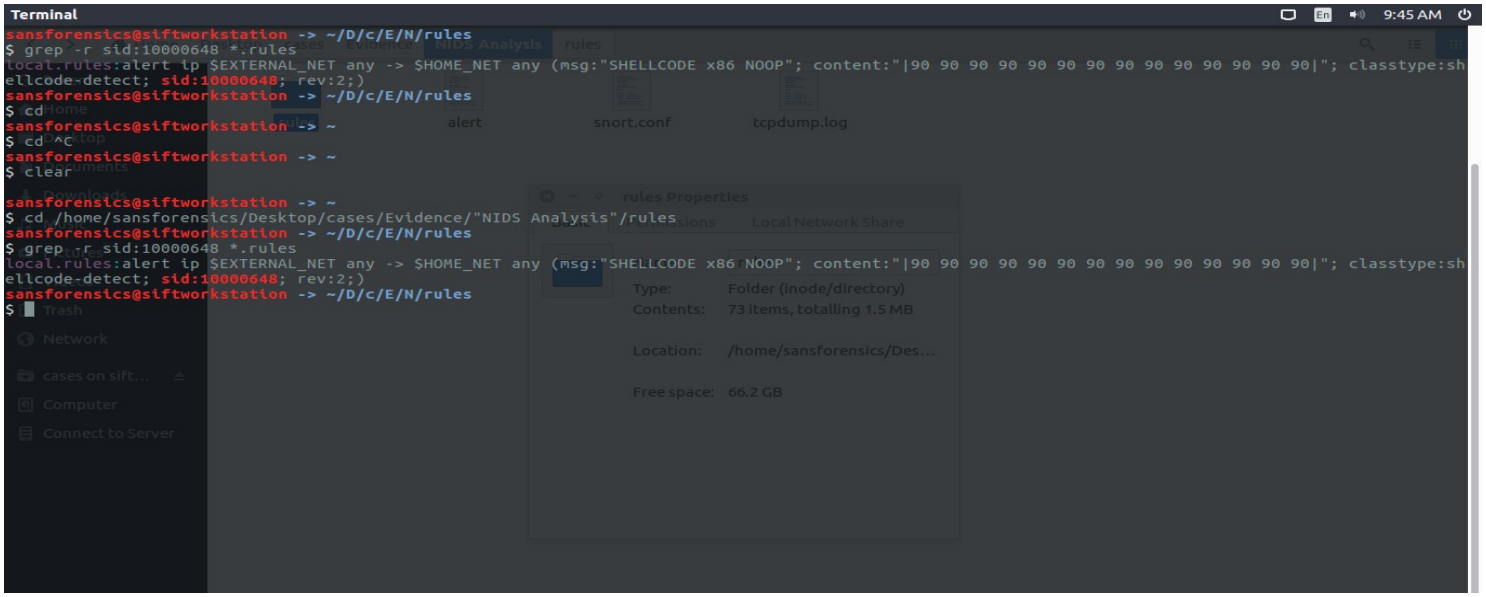

Figure 1.2.1

1.3 Retrieve the packet that triggered the alert

Figure 1.3.1 and 1.3.2, we open the **tcpdump.log** file in wireshark and use the filter **ip.id=-53309** to analyse the packet and we were able to find **Seq: 0x1B2C3517** and **Ack: 0x9F9E0666** within TCP hex dump.

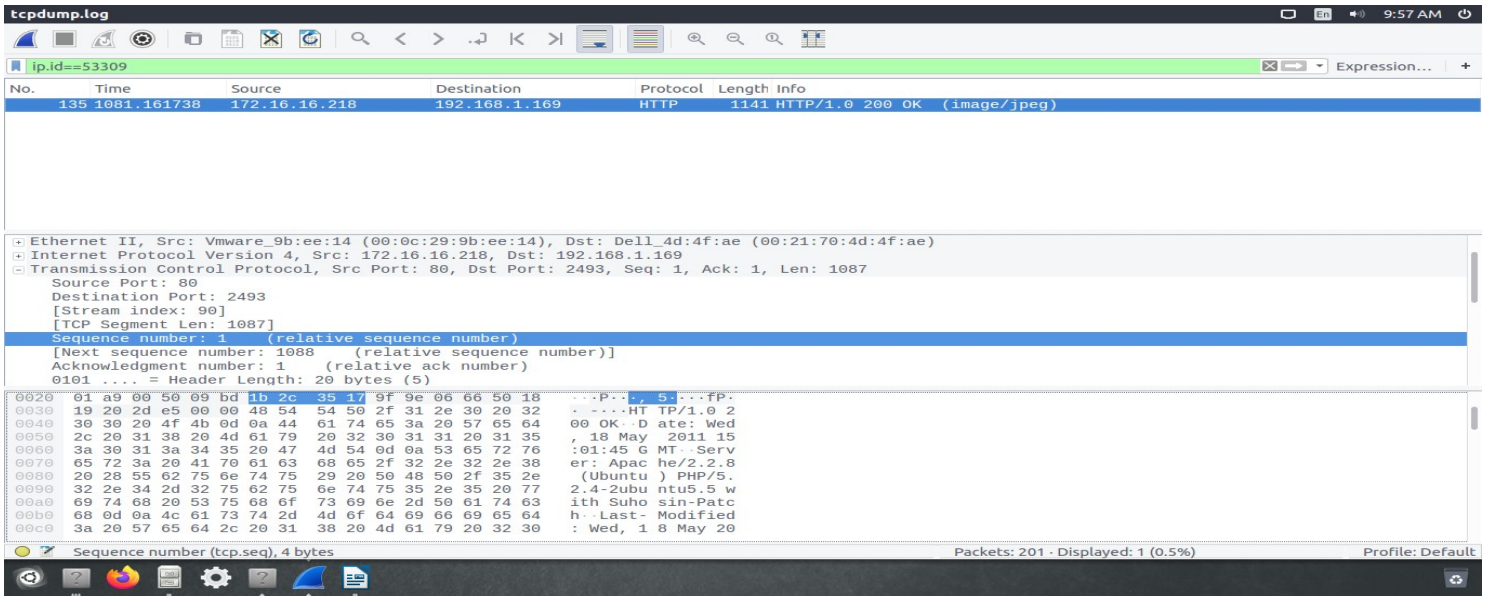

Figure 1.3.1

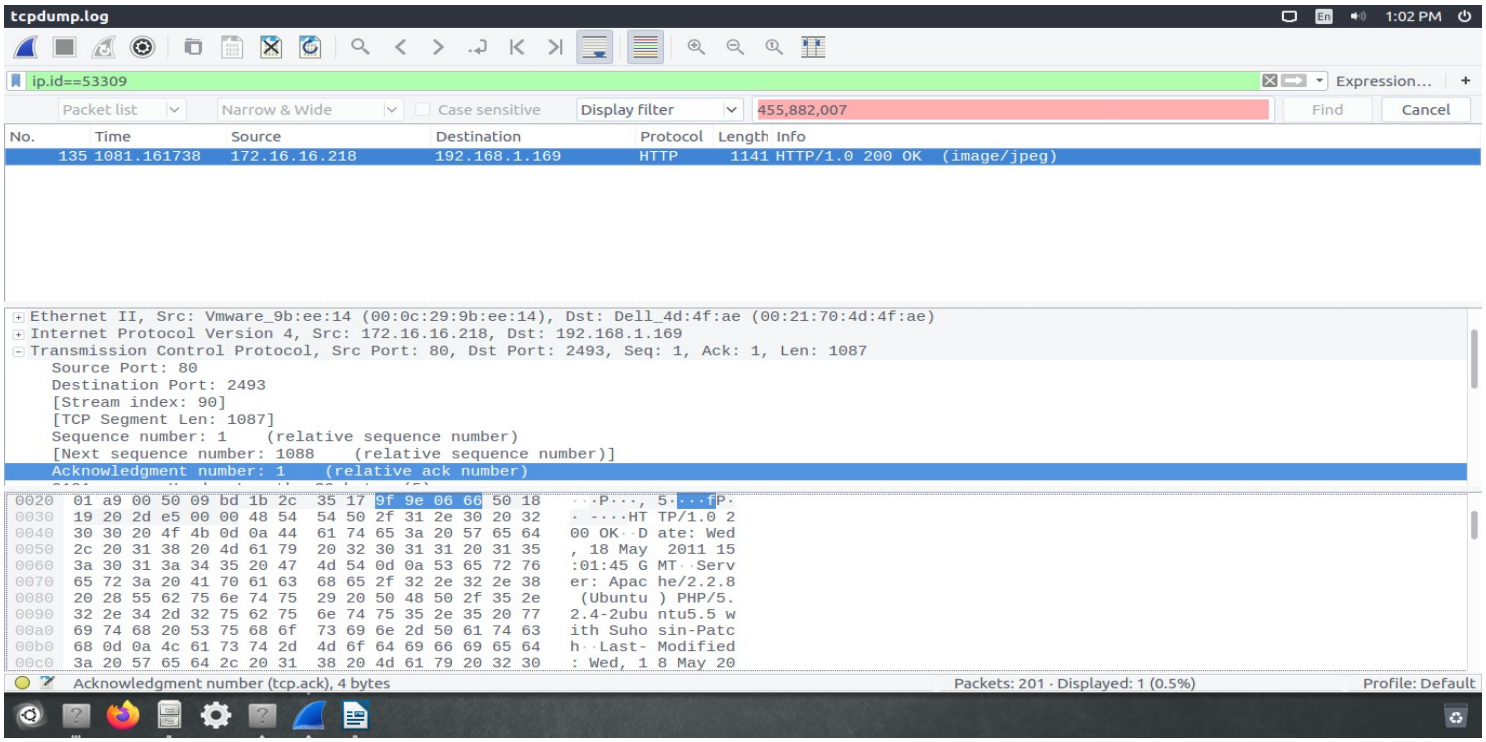

Figure 1.3.2

1.4 Compare the rule to the packet to understand WHY it fired

Figure 1.4.1 and 1.4.2, we compare the rule and packet and find that the rule is set to detect suspicious content from an external IP to port 80 and here an image file is being transferred with potentially malicious shellcode.

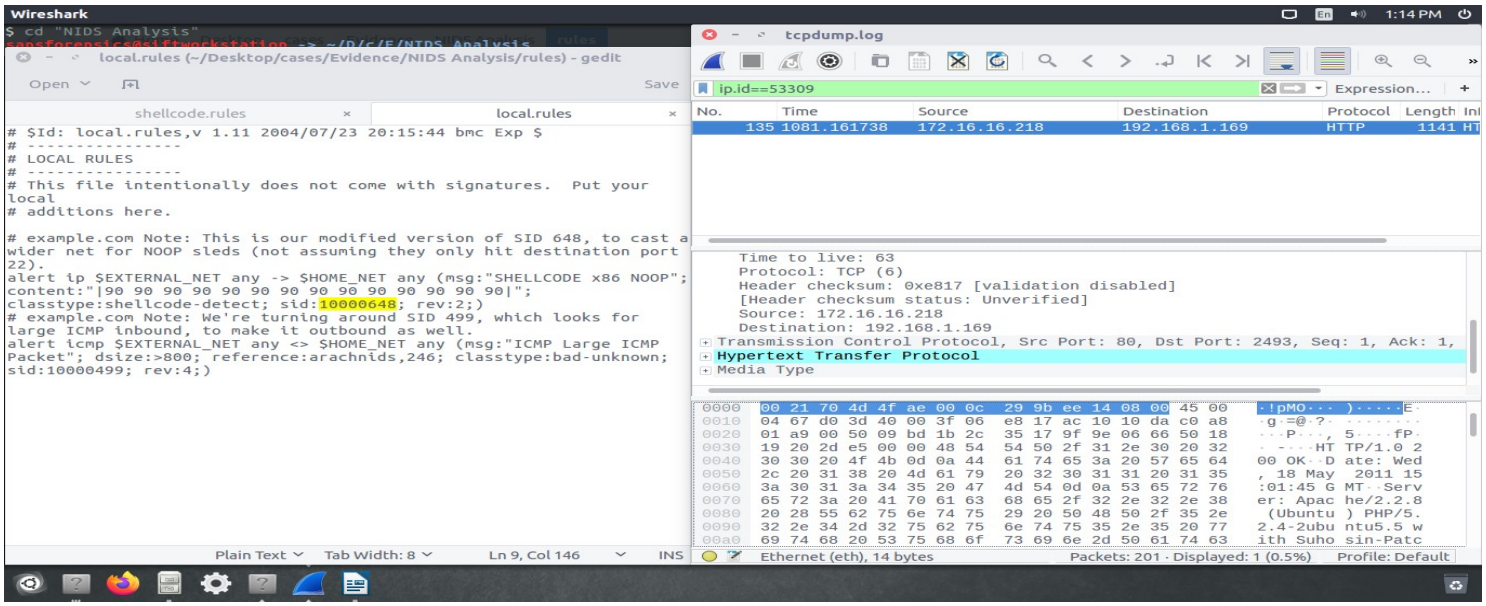

Figure 1.4.1

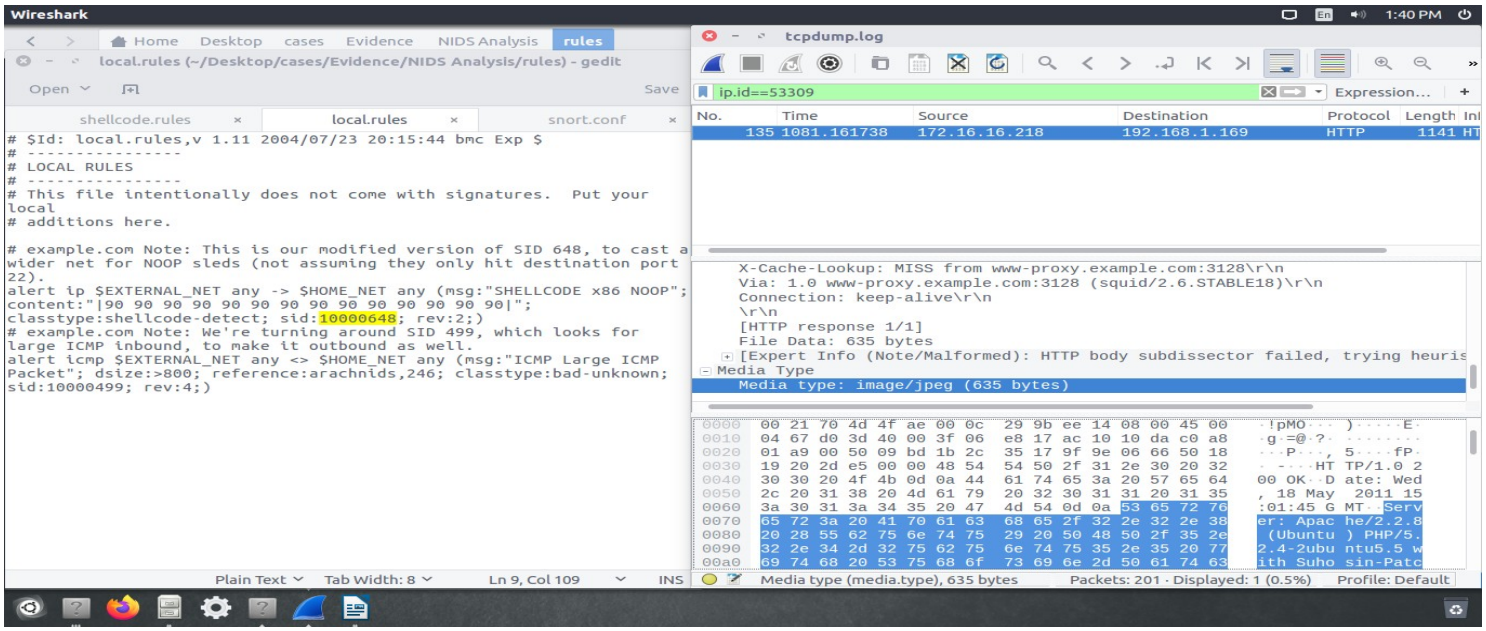

Figure 1.4.2

1.5 Construct a timeline of alerted activities involving the potentially malicious outside host

In Figure 1.5.1, 1.5.2, 1.5.3 and 1.5.4:

- at 07:43:51 we detect “COMMUNITY MISC BAD-SSL tcp detect [**] [Classification: Misc activity]” from 204.11.50.137:80 to internal host 192.168.1.169
- From 07:43:52 to 08:15:08, we detect “INFO web bug 0x0 gif attempt [**] [Classification: Misc activity]” from different external hosts to internal hosts like 192.168.1.169, 192.168.1.170etc
- In between the above time frame, at 08:01:45 MST we also detect “SHELLCODE x86 NOOP [**] [Classification: Executable code was detected]”, from an external web server 172.16.16.218:80 to 192.168.1.169

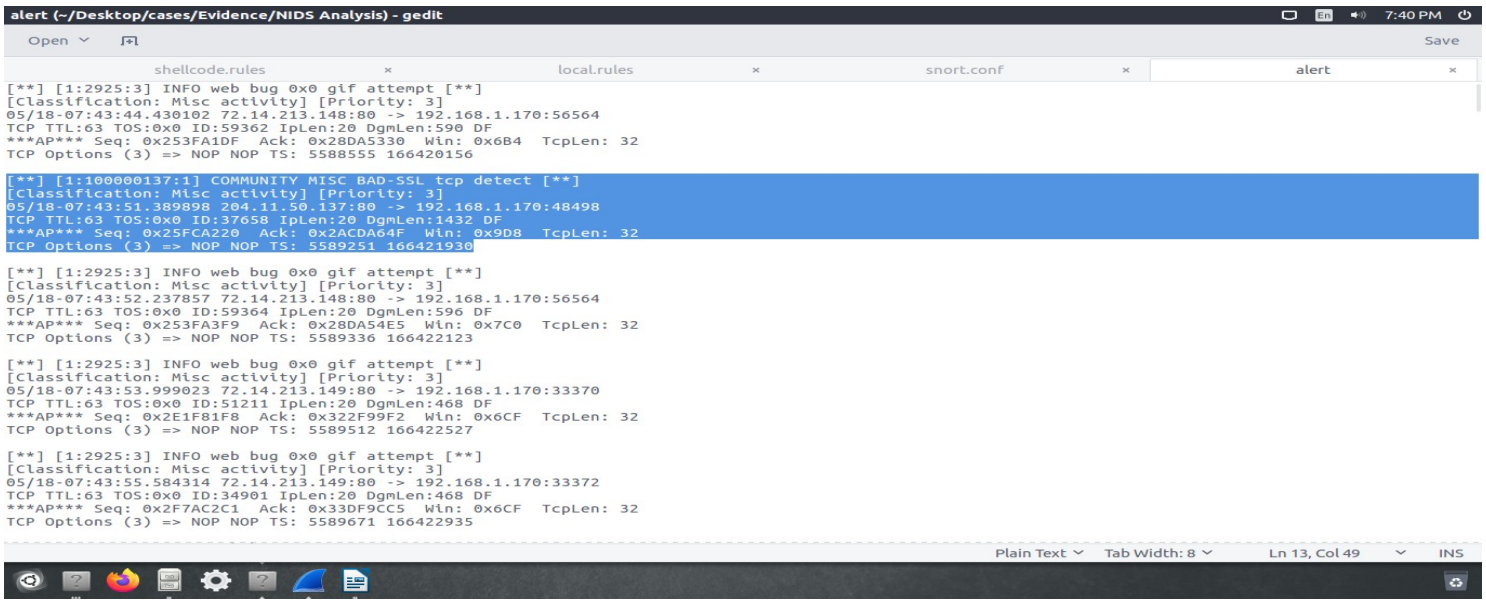

Figure 1.5.1

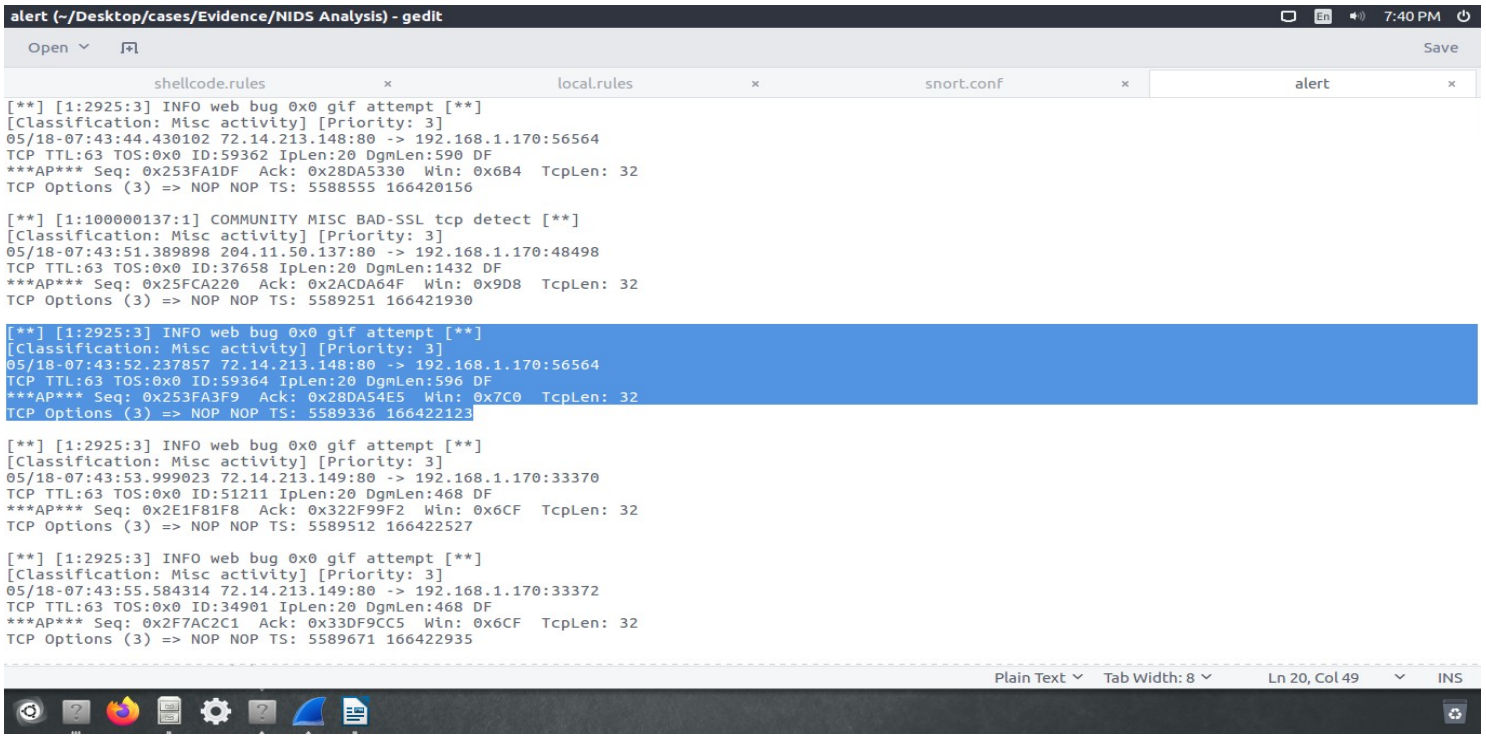

Figure 1.5.2

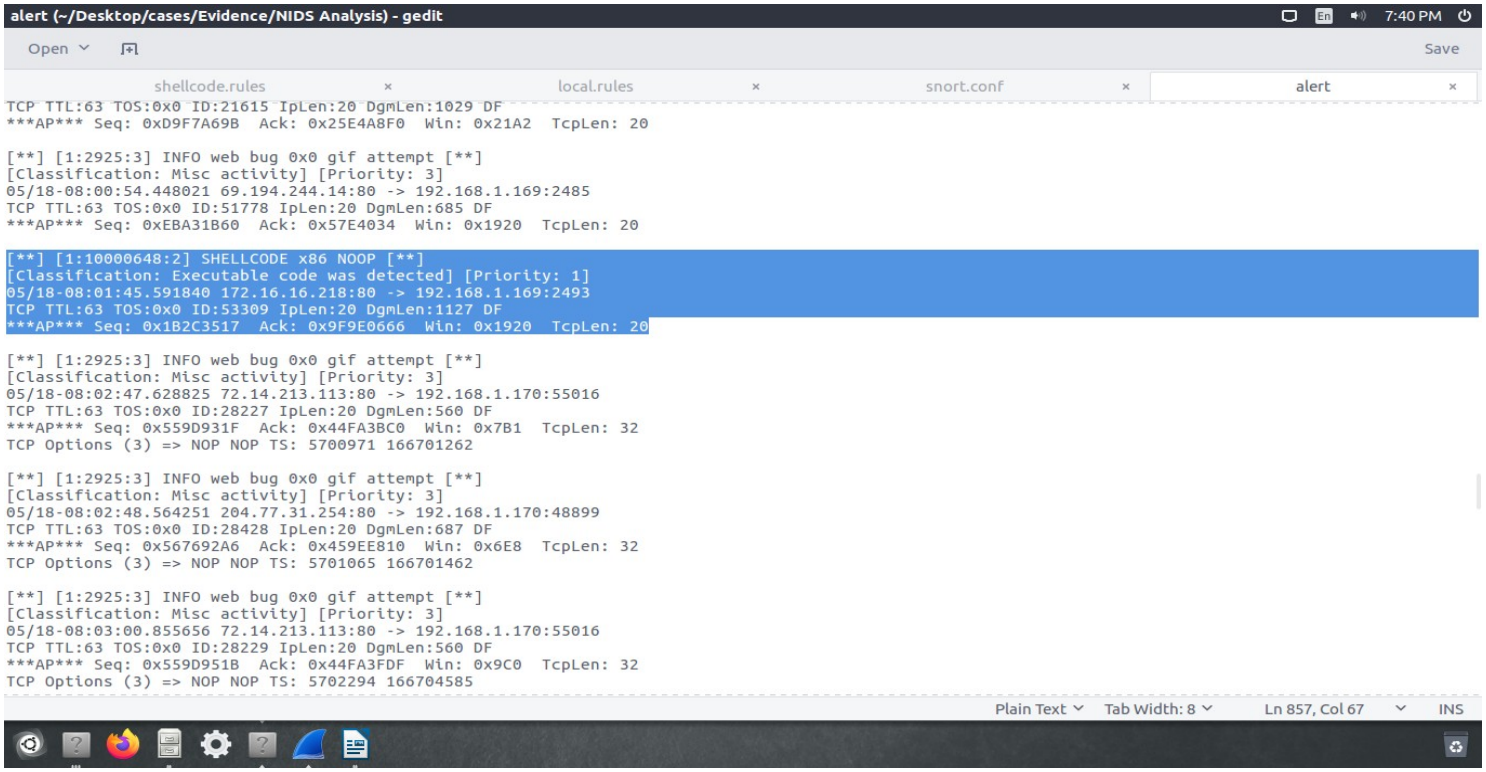

Figure 1.5.3

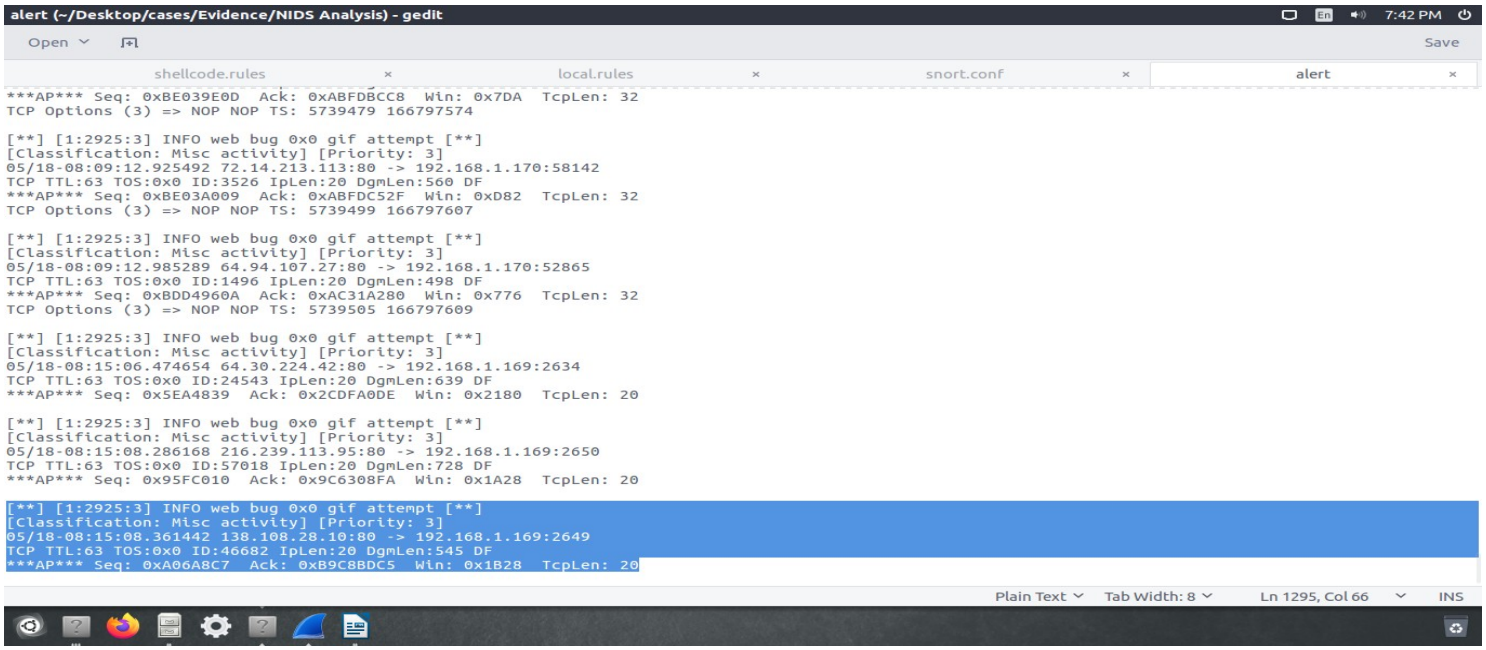

Figure 1.5.4

1.6 Construct a timeline of alerted activities involving the target

Figure 1.6.1, 1.6.2:

- at 08:01:45 MST we detect “SHELLCODE x86 NOOP [**] [Classification: Executable code was detected]”, from an external web server 172.16.16.218:80 to 192.168.1.169. This signifies the malicious intent to inject shellcode.
- and at 08:15:08, we detect “INFO web bug 0x0 gif attempt [**] [Classification: Misc activity]” from different external hosts to internal host 192.168.1.169. This type of alert was also found at many other instances.

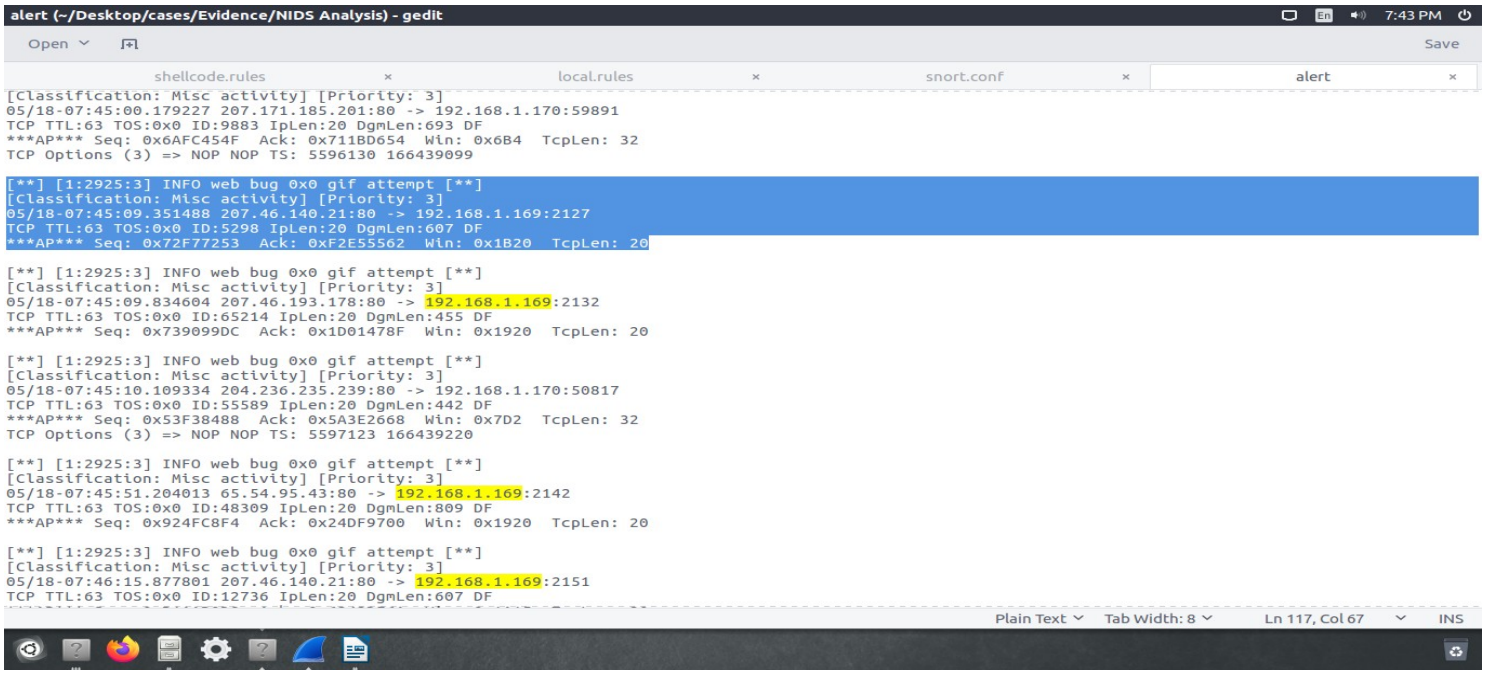

Figure 1.6.1

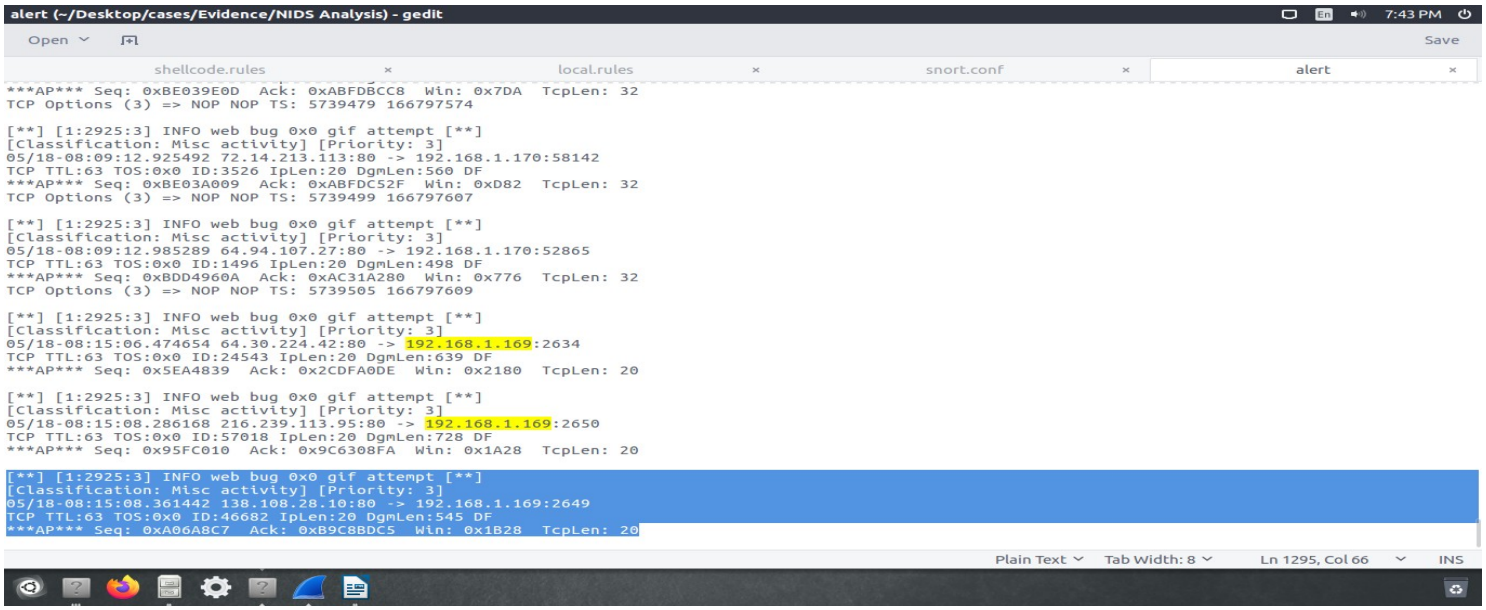

Figure 1.6.2

2. Proxy Log Background (Part 2)

2.1 Determine whether the evidence extracted from the Squid cache corroborates our findings from the Snort logs.

Figure 2.1.1, an unusual binary sequence that is commonly associated with buffer overflow exploits was found in ETag in the external webserver's HTTP response : **1238-27b-4a38236f5d880.**

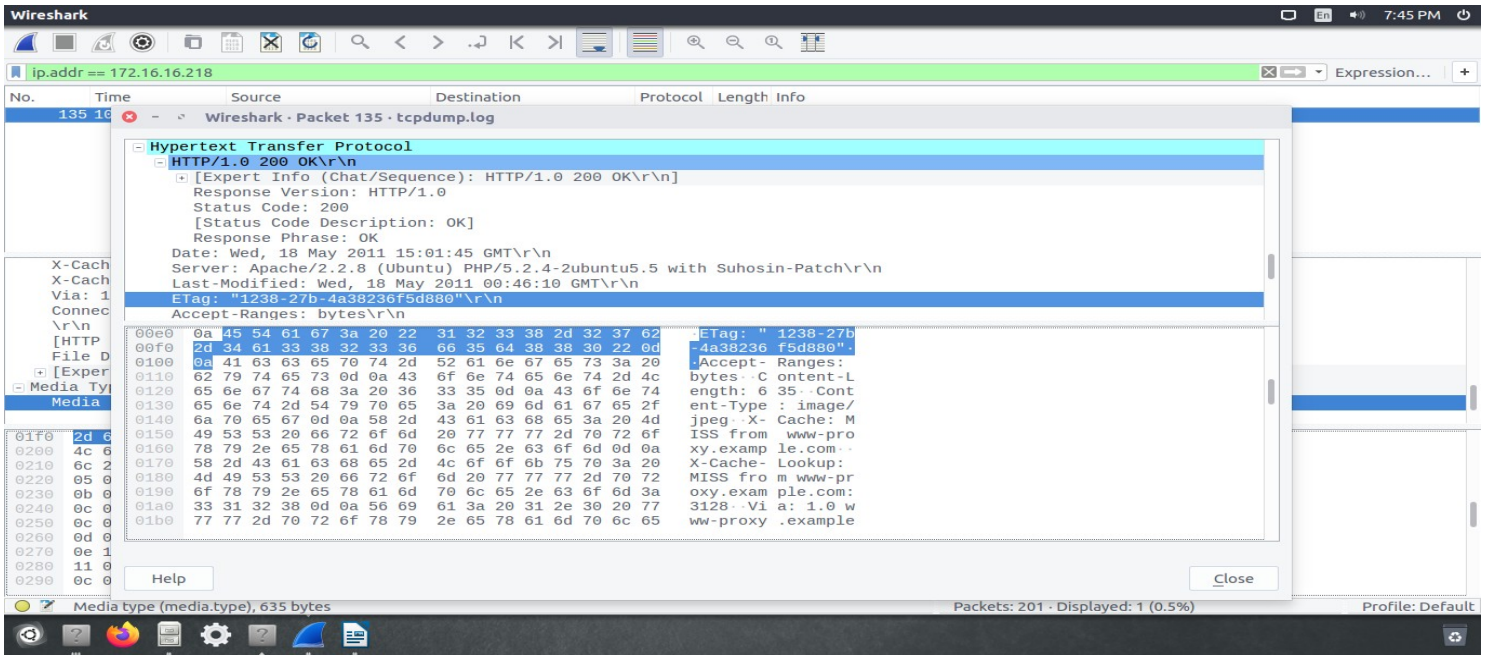

Figure 2.1.1

Figure 2.1.2, we use **cd /home/sansforensics/Desktop/cases/Evidence/"Proxy Log Analysis"/squid** and then **grep -r '1238-27b-4a38236f5d880'** to look for file with the matching ETag sequence and we find that a binary file with location "00/05/0000058A" has matched our search.

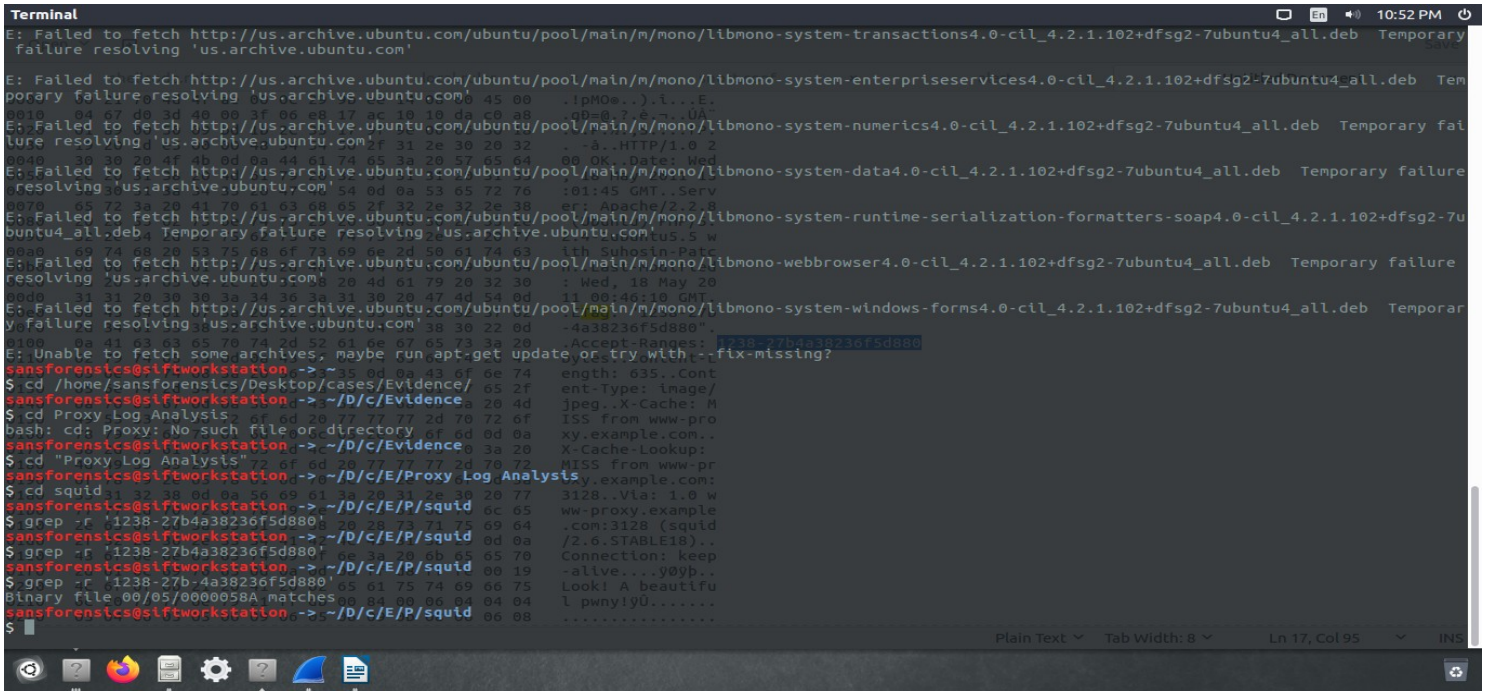

Figure 2.1.2

Figure 2.1.3, we open the binary file 0000058A, using Bless Hex Editor and could verify the matching ETag sequence - **1238-27b4a38236f5d880**. We were also able to find the location **http://www.evil.evl/pwny.jpg** of the malicious JPEG file, **pwny.jpg** in ASCII characters.

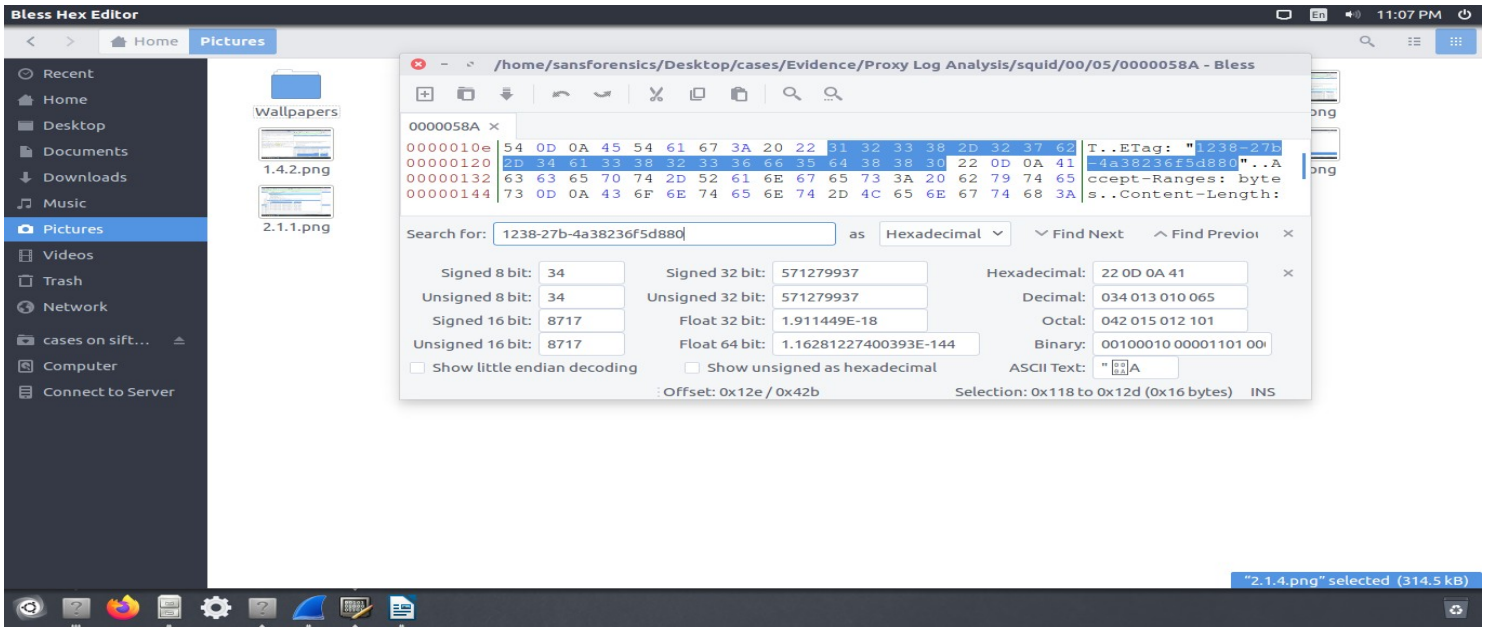

Figure 2.1.3

Figure 2.1.4, using **cd /home/sansforensics/Desktop/cases/Evidence/"Proxy Log Analysis"/squid** and then **grep -r "http://www.evil.evl/pwny.jpg"** we search for the .jpg file's url among other cache files in the squid directory and find two binary file matches **0000058A** and **00000589.**

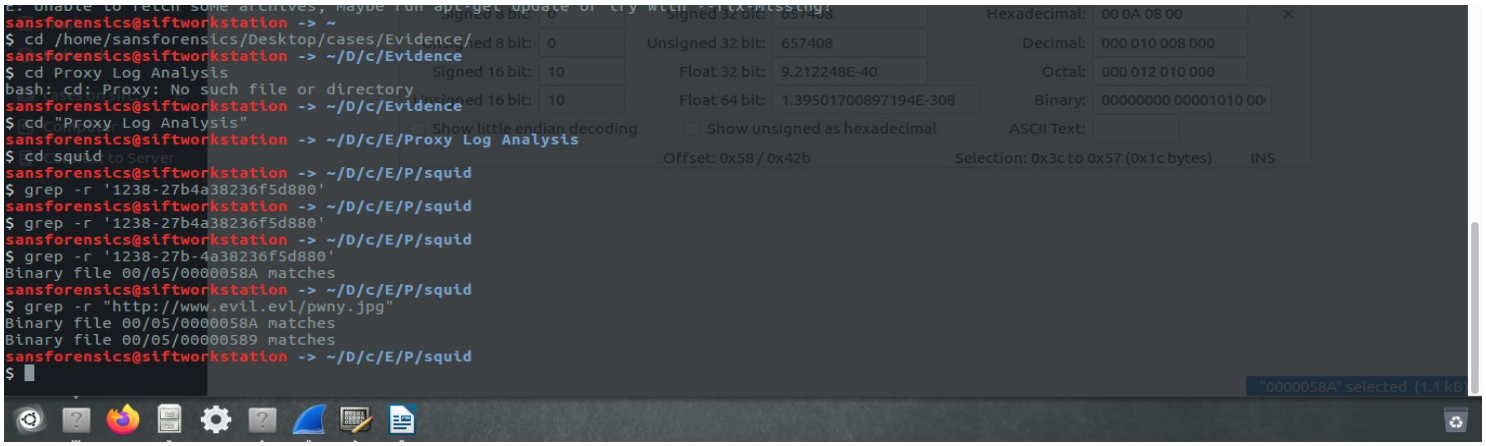

Figure 2.1.4

2.2 Present any information you can find regarding the identity of any internal users who have been engaged in suspicious activities.

Figure 2.2.1 and 2.2.2, we copy the **access.log** and **store.log** files from **var-log-squid.zip** to a Windows machine to open them using MS Excel and filter them using 192.168.1.169. Then we hit and try different epoch timestamps to match with the alert timestamp. For this we use an online converter (**figure 2.2.3**) and find out that **1305729906**, **1305730906** and **1305731047 (blue color coded)** are the closest timestamps corresponding to **07:45:06** (when internal host 192.168.1.169 started browsing external web sites)**, 08:01:46** (when alert triggered for shellcode injection) and **08:04:07** (when internal host 192.168.1.169 sent crafted packets to other internal hosts).

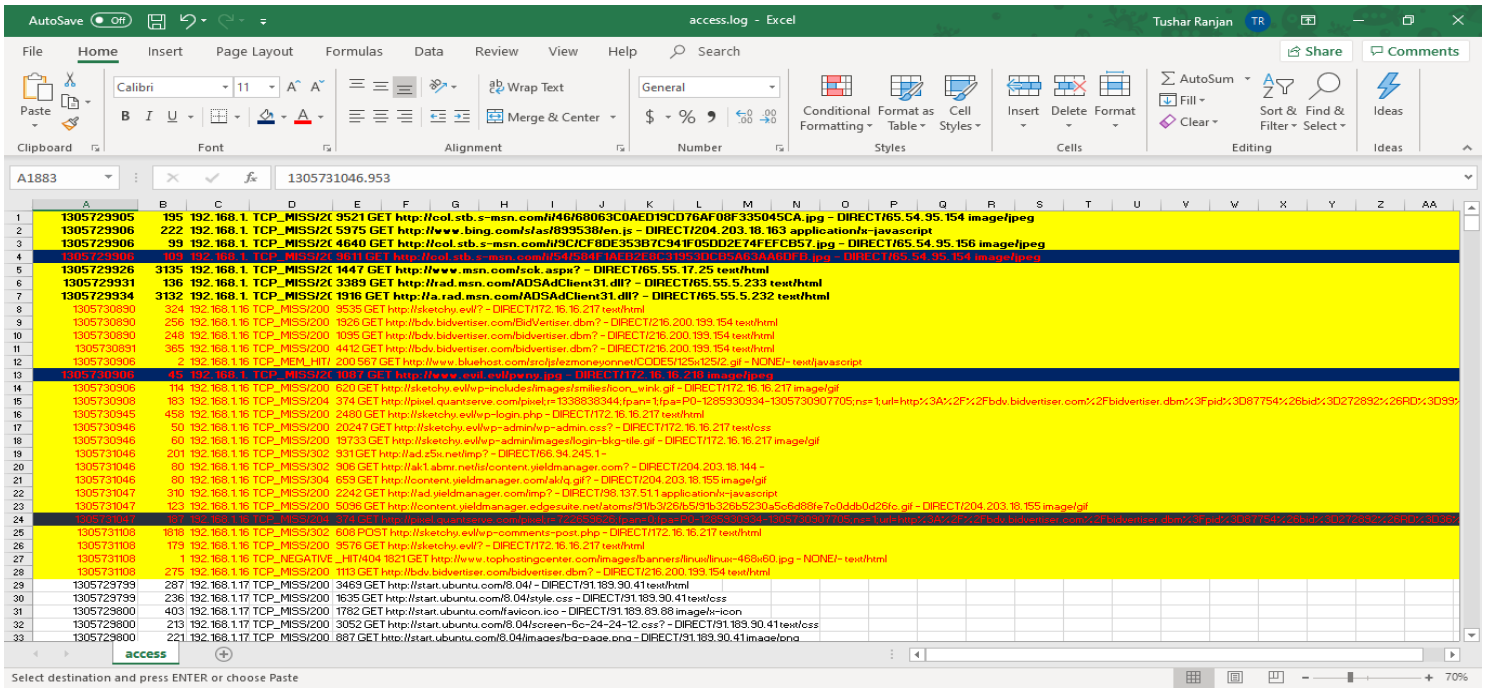

Figure 2.2.1

After careful analysis of activities from both access.log and store.log files around the aforementioned timestamps, we find out that the internal user was browsing images on various external websites, among which also happened to show some activity with .evl websites like **http://sketchy.evl/?, www.evil.evl/pwny.jpg** (.evl is a top level domain used by Evil Systems). This user also seems to have admin access to one of the .evl website because there are traces of login attempts to admin profiles.

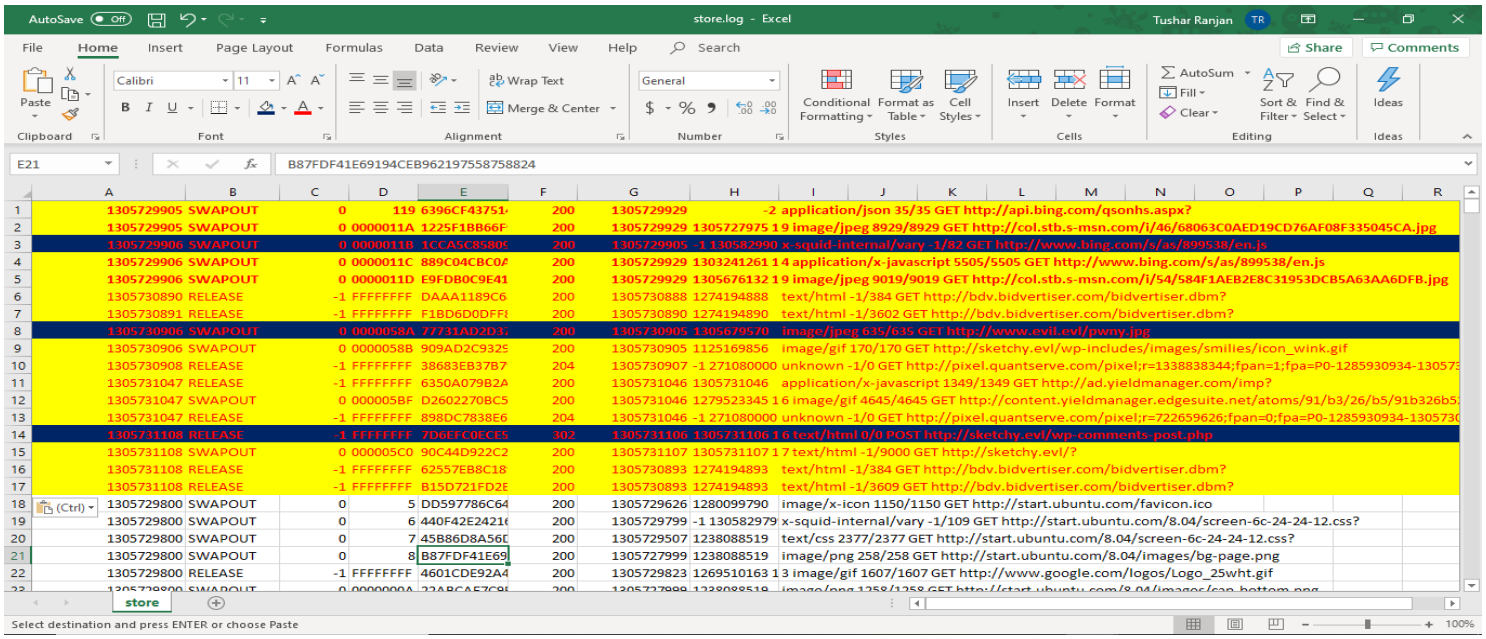

Figure 2.2.2

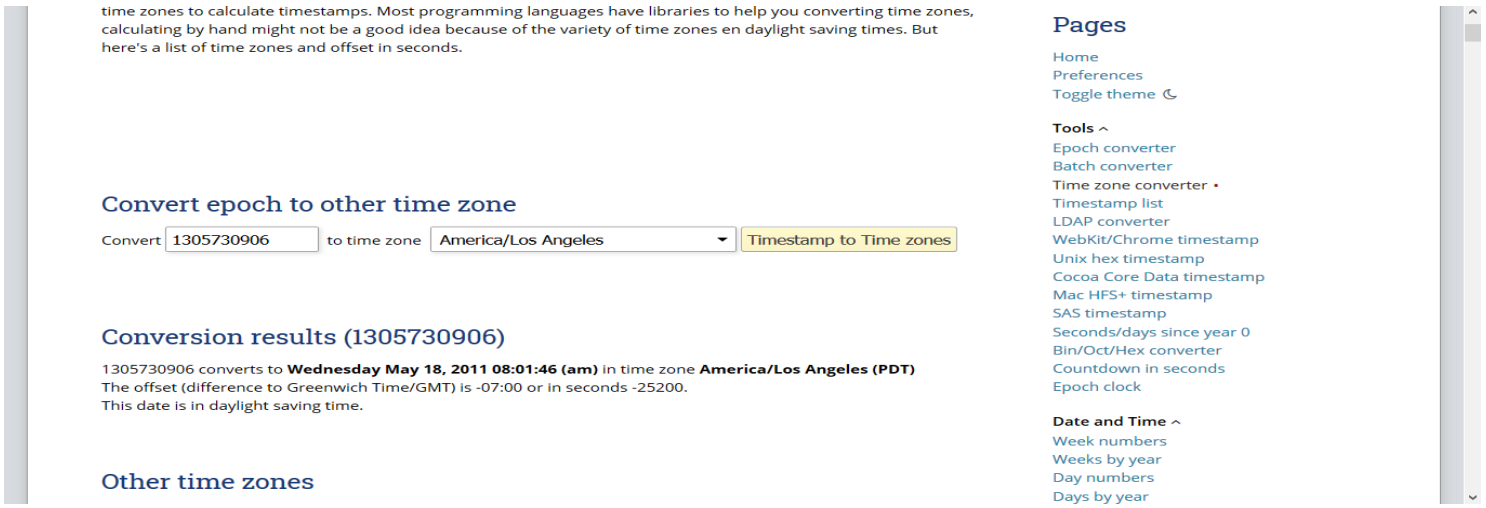

Figure 2.2.3

Conclusion

The NIDS/NIPS and Web Proxy Analysis lab taught me what tools I can use to to to a) Perform log analysis on IDS/IPS alerts (Snort) to become familiar with alert formats and using them to investigate potentially malicious traffic and b) Perform log analysis on Proxy logs (Squid) to become familiar with proxy log formats, and how to review them to investigate web related traffic. This lab also gave me a good idea about how real-world digital evidences are collected and preserved for future use to dig deeper to investigate potential case of an insider threats just by capturing their logs and network activity.

After the analysis I was able to infer that, an insider from the company is using the internal host to browse images on various external sources, with one of them being a highly malicious domain. There were also the traces of admin level logins to .evl domain leading to a malicious .jpg file trying to inject shellcode.

Glossary

1. Network Intrusion Protection Systems (NIPS) and Network Intrusion Detection Systems (NIDS) are tested on the Technologies and Tools portion of the Security+ certification exam. This article details what is covered on the Security+ certification exam regarding these important network security devices. This article should not substitute for studying but rather serve as a brief review and guide for areas that you may need to look over again.
2. Firewall Analyzer (Proxy Log Analyzer) collects and archives the proxy server logs, analyzes them, and generates useful corporate internet access information reports. Proxy server reports provide network security administrators and managed security service providers (MSSP) with important insight into the efficiency of their corporate Internet usage. As a proxy log analysis tool, Firewall Analyzer supports BlueCoat, Microsoft ISA, Squid proxy logs and servers.
3. Wireshark is the world's foremost and widely used network protocol analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions. Wireshark development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by Gerald Combs in 1998.
4. grep is a command-line utility for searching plain-text data sets for lines that match a regular expression. Its name comes from the ed command g/re/p, which has the same effect: doing a global search with the regular expression and printing all matching lines.
5. Snort is an open source network intrusion detection system (NIDS) created by Martin Roesch. ... Through protocol analysis and content searching and matching, Snort detects attack methods, including denial of service, buffer overflow, CGI attacks, stealth port scans, and SMB probes.

References

[1] “Technologies And Tools - NIPS / NIDS.” Infosec Resources, resources.infosecinstitute.com/category/certifications-training/securityplus/sec-domains/technologies-and-tools-in-security/installing-and-configuring-network-components-to-support-organizational-security/technologies-and-tools-nips-nids/.

[2] Hoffman, Chris. “How to Use Wireshark to Capture, Filter and Inspect Packets.” How, How-To Geek, 14 June 2017, www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/.

[3] “Grep Command in Unix/Linux.” GeeksforGeeks, ., 20 May 2019, www.geeksforgeeks.org/grep-command-in-unixlinux/.

[4] Rouse, Margaret. “What Is Snort? - Definition from WhatIs.com.” SearchMidmarketSecurity, TechTarget, 21 Sept. 2005, searchmidmarketsecurity.techtarget.com/definition/Snort.